Seminar 1,
Conceptual Model

Data Storage Paradigms, IV1351

Esra Salman

esras@ug.kth.se

9/11 – 2023

Table of Contents

1 Introduction

Seminar 1 is the initial phase in developing a single database application. The objective of this project is to create a database for the *Soundgood music school co* for information handling and business transaction facilitation, including all data and several specified operations. The foundation of the school is built on selling music lessons to students, which are offered in various categories and levels. The intended learning outcomes are stated as follows: “Model needs for information based on an organizational description and convert the model to a functioning database”.

With the objective to practice conceptual modeling, the preparation of this seminar included reading Ch. 3 in the provided course literature, “Fundamentals of Database Systems” alongside the lectures in Conceptual Model module in canvas. Covering the subjects of creating a Domain Model, IE notations, and Conceptual Model parts 1 & 2.

The main objective of this seminar is to create a *Conceptual Model*, in this case an in this specific case *entity-relationship* model (ER), to visualize the various data entities concerned. Conceptual modeling, is used for conceptual design of database applications. It is also a model representing parts of reality to be stored in a database, exempt from database entities. It describes the abstract representation of reality and how its data is organized and interconnected. To create the *ER* model a UML, *Unified Modeling Language*, methodology was implemented using *Astah Professional* application.

The seminar tasks are dual, first part mandatory to create a conceptual model for the school database, and the second is a higher-grade task to implement inheritance in the model diagram. Concerning for the former, all data provided in the description is to be included. The information provided contains data on the student, instruments, skills, lessons, instructors, price ranges, salaries, payments, and renting system for instruments.

This seminar task was made alongside Daniel Ibrahimi and Ermia Ghaffari

2 Method

To start this seminar, an extensive preparation was made by thoroughly examining Ch. 3 in the course literature, " Fundamentals of Database Systems " as well as a comprehensive review of lectures from Conceptual Model.

As mentioned above, seminar 1 primarily consists of designing an ER model. In accordance with the requirements, decisions were made in modeling in regards to the various data provided. The UML modeling application *Astah professional* was used for both tasks. The first started by brainstorming possible entities, realized with methodologies as *noun identification* when listing the data provided in the description. This method eludes a noun singular, in this case entities as student, instructor, instrument etc. These represent real-life independent existents, with a physical or conceptual existence.

Thereafter with the help of the *category list* more entities were eluded, those include *transactions, products or services, roles, places, records of a transaction, events, physical objects, devices, descriptions, catalogs, systems, quantities and units*, and *resources.* In the last part of this step is to remove all unnecessary categories,

Following the identification of entities, the next step is an iteration of three parts, 1. Attributes, 2. Relations and cardinalities, 3. Implement user-like mind for the user interface to be of relevance. After the derivation of entities, their attributes are listed individually. Attributes are descriptive properties belonging to the specified entity, each attribute has a value connected to its entity, i.e., age of fifty. Attributes could include *studentID* for the entity *Student* and so on.

In ER models, there are various attributes, such as simple or composite or NULL. These are relevant for future seminars. Attributes also have different criteria, they must either be strings, Boolean, number, time, and should not have attributes, neither is it part of an entity.

Afterwards, relations between entities are drawn. To draw the correct cardinality, the difference between identifying and non-identifying relationships are of importance as they define how entities relate to each other. The former is relevant when an entity is dependent on the other, making it a child entity and the latter a parent entity making it identifiable and dependent existent. The non-identifying relationship is the opposite, which is when there is no dependency between entities. This implies that entities are independent and do not rely on any other entity.

For the second task inheritance was implemented. This concept refers to the allowance of inheriting attributes and relationship between entities. For the inheritance concept to become a reality in a conceptual model, hierarchical entities are listed from the *Superclass* to the *Subclass*. The former acting as the inherited and the latter as the inheritor, also child-class (read above section).

3 Result

The Conceptual Models are published in GitHub:

https://github.com/esrsal/IV1351.git

Conceptual Model:

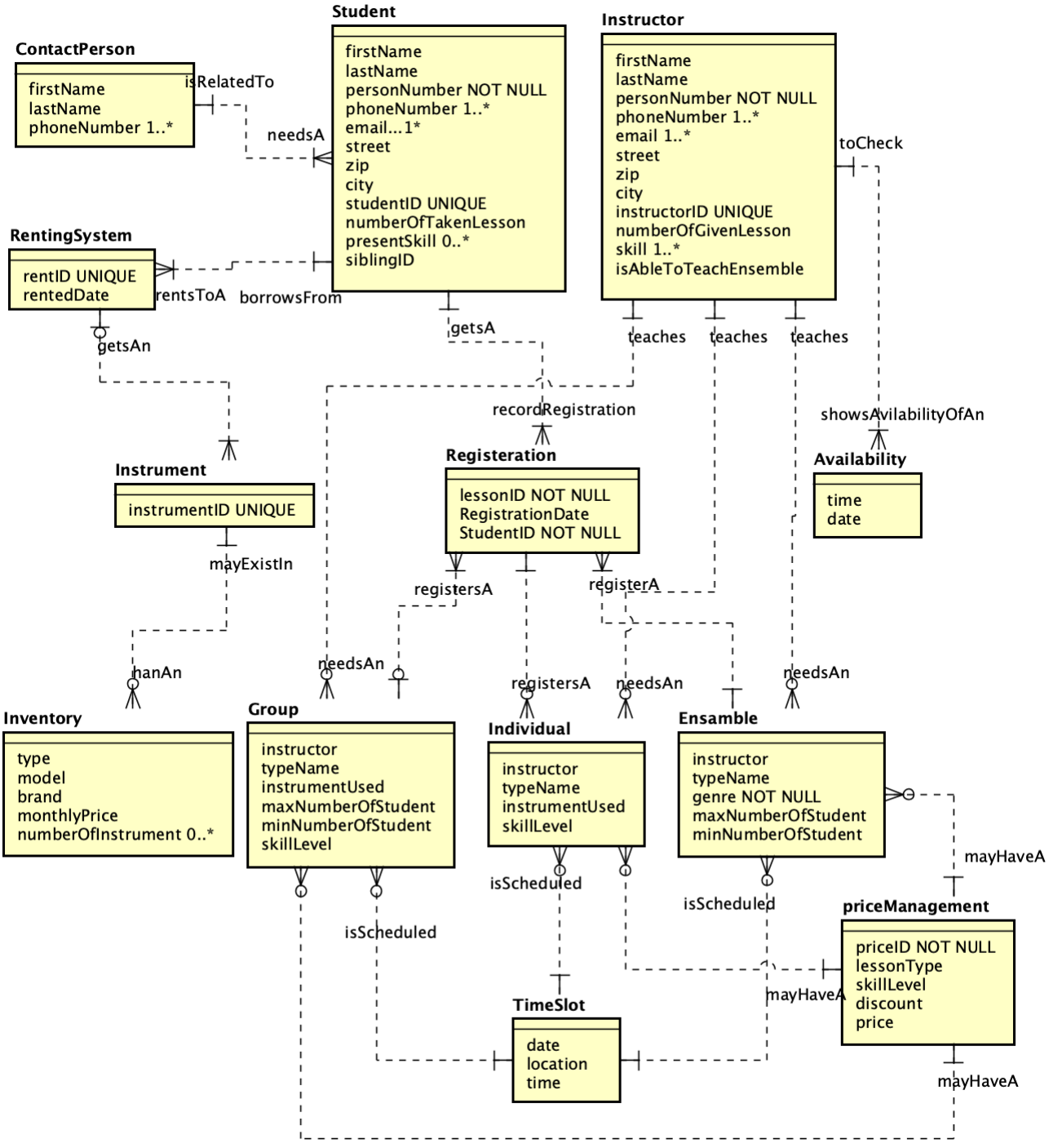

Figure 3.1: Conceptual Model of all relevant entities, attributes, and relations.

In fig 3.1 a visual representation of the Conceptual model without inheritance is shown. The entities are given their attributes, such as *Student* given *studentID UNIQUE, presentSkill,* and *siblingID* and so on. Also, relations between entities are drawn, *Student* have a relation with *ContactPerson,* which reads as "Student needs a contact person", or the other way around "a contact person is related to one or many students" (the triple end of the relation-line). The three lesson versions are displayed, *Group, Individual,* and *Ensemble.* The model was designed in a way to simplify the various entity-categories, with actual persons at the top, followed by smaller attribute-entities such as the objects *Instrument, Registration, RentingSystem,* and *Availibilty,* then larger entities in the lower part.

In fig. 3.2 the visual representation of the Conceptual Model with inheritance is shown. Various attributes in the model above have been moved to a *Superclass* in the inheritance model, for instance attributes from both *Student* and *Instructor* are moved to a new entity called *Person*, where both the subentities now inherit. The same goes for the now *Superclass Lesson*, which the three kinds of lessons *Group, Individual,* and *Ensemble* inherit.

Conceptual Model with inheritance:

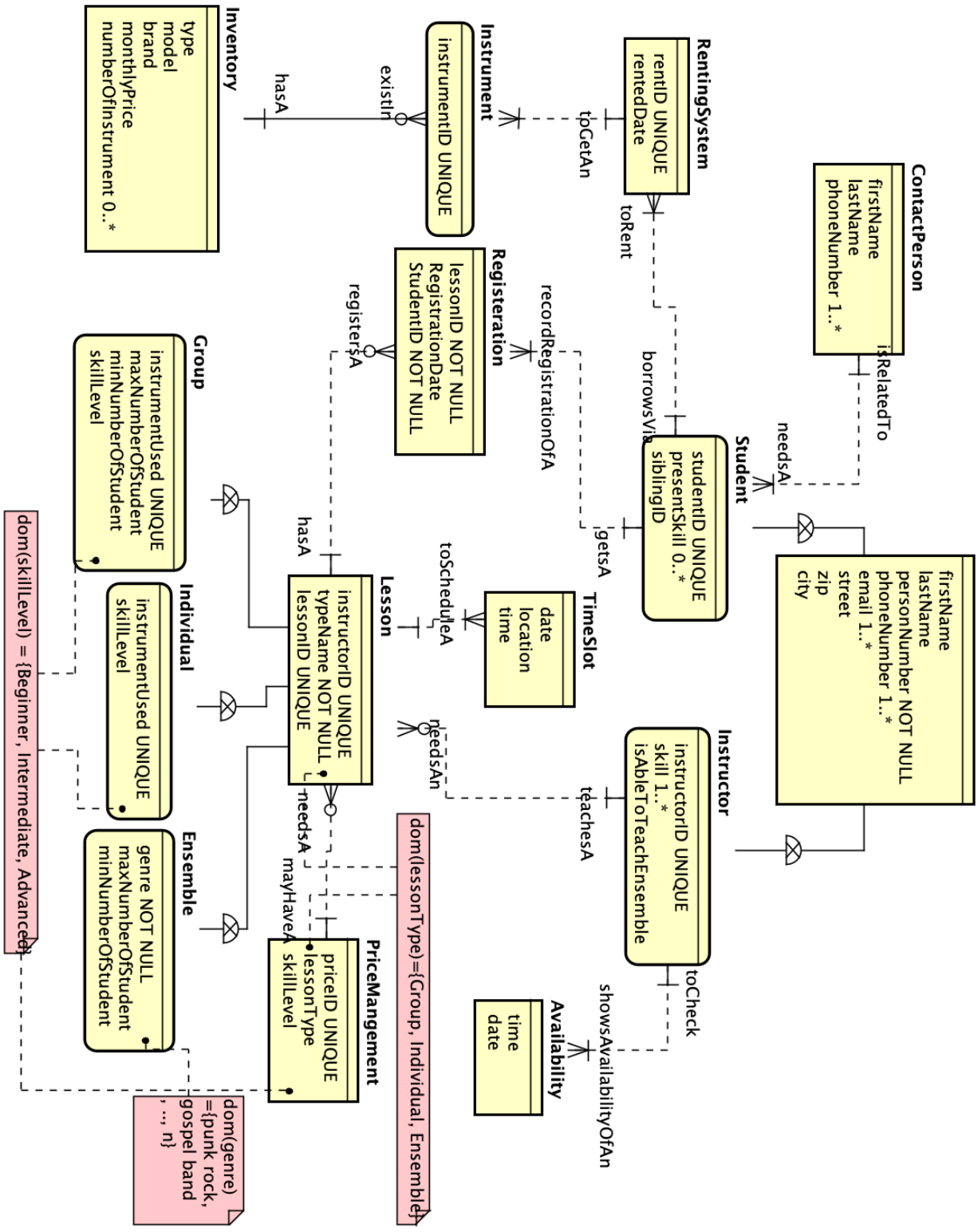

Figure 3.2: Conceptual Model of all relevant entities, attributes, relations, and inheritance.

4 Discussion

The Conceptual Models presented in this seminar are designed with all requirements taken into consideration, as mentioned in section 2. In the requirements description the type of roles, objects, and systems are provided, all of which are included as entities in the model. Moreover, attributes and relations are drawn, discussed, reevaluated, and confirmed with the teacher, to ensure their correction. All entities also have attributes as it seemed reasonable, even the smallest such as *Instrument.*

Further, the attributes are included with the given criteria mentioned in section 2. As for the cardinality, i.e., NOT NULL and/or UNIQUE, are given to the relevant attributes such as *InstrumentID,* since it has a unique and non-duplicate ability. In other attributes, as *typeName* is given the cardinality *NOT NULL as a reasonable measure.* There are also noted comments for cardinalities of attributes to clarify their use, see fig. 3.1.

Moreover, relations are drawn between entities with their importance considered and unnecessary ones removed. Nonetheless, the length of some relations could be lessened. Also, naming conventions are followed through as per the requirements and made clear, such as *PriceManagment, isAbleToTeachEnsemble,* and so on, alongside correctly used notation in UML.

Finally, the inheritance implementation in fig 3.2 is used in multiple places, for instance in *Person* and *Lesson.* Fig 3.1 shows the CM model without any inheritance, as explained in section 3. There are both advantages and disadvantages in using inheritance, those include the coherency of the model when using it. The model has a better overview, easier to follow entities, due to *Superclass* connecting various subclasses, making it more flexible. The number of attributes is not duplicated, as mentioned in section 2 and 3. Nonetheless, the disadvantages also include some kind of inflexibility as the dependencies between classes are greater. The relations between entities are also increased, making the model harder to comprehend and facilitate.